VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Ville Sippola

# LABYRINTH

Kiihtyvyysanturiin perustuva peli Android-laitteille Unity 3D-pelimoottorilla

Tekniikka

2017

VAASAN AMMATTIKORKEAKOULU

Tietotekniikka

# TIIVISTELMÄ

| | |
|---|---|
| Tekijä | Ville Sippola |
| Opinnäytetyön nimi | Kiihtyvyysanturiin perustuva peli Android-laitteille Unity 3D-pelimoottorilla |
| Vuosi | 2017 |
| Kieli | suomi |
| Sivumäärä | 38 |
| Ohjaaja | Timo Kankaanpää |

Sain idean päättötyöhöni jo Vaasan ammattikorkeakoulun toisella luokalla ollessani kun meillä oli JavaScriptiä. Halusin tehdä pelin ja silloinen luokanvalvojani Pirjo Prosi mainitsi, että peli aiheena kelpaa hyvin, mutta alustan olisi hyvä olla joku muu kuin PC. Muistin isoäidillä vanhan mekaanisen labyrinttipelin ja ajattelin, että senhän voisi toteuttaa puhelimelle ohjauksen perustuessa puhelimen kiihtyvyysanturiin, eli ohjaus toimisi kallistamalla puhelinta. Päättötyön ensisijainen tarkoitus on oppia ohjelmoimista ja Unity 3D:n käyttöä.

Pelin idea on ohjata palloa labyrintissä samalla keräten pisteitä sekä välttelemällä ansoja. Sen voi läpäistä pääsemällä lipun luokse. Peli on suunnattu lapsille ja myös vanhemmillekin satunnaispelaajille. Alustana toimii Android-laitteet, joskin olen testannut tätä paljon PC:llä.

Koko peli on toteutettu Unity 3D-pelimoottorilla ja skriptauskieli on C#. Suuri osa peleissä olevista objekteista on ladattu Unity Asset Storesta ja käytetty Googlea paljon tiedon, tekstuurien sekä äänien hakemiseen. On katsottu myös paljon tutorial-videoita Unityn sivuilta ja Youtubesta. Muita käytettyjä ohjelmia ovat Microsoft Visual Studio skriptien editoimiseen ja Blender 3D-mallinnukseen.

Pelin perusidea toimii hyvin, mutta kaupallistamiseen on vielä matkaa. Lopullinen peli aiotaan julkaista Google Play Storessa.

| | |
|---|---|
| Avainsanat | Unity 3D, C#, Android, kiihtyvyysanturi, peli |

VAASA UNIVERSITY OF APPLIED SCIENCES

Information technology

# ABSTRACT

| | |
|---|---|
| Author | Ville Sippola |
| Title | Game for Android-devices, where controls are based on the use of velocity sensors, made with Unity 3D game-engine. |
| Year | 2017 |
| Language | Finnish |
| Pages | 38 |
| Name of Supervisor | Timo Kankaanpää |

I got the idea for my thesis when I was at the second grade in Vaasa university of applied sciences where we had JavaScript. I wanted to make a game and our current class supervisor Pirjo Prosi mentioned, that game as subject will do fine, but the platform should be other than PC. I remembered my grandmothers old mechanical Labyrinth-game and thought I could make that to phone and base the controls to velocity sensor, so the controls would work by tilting the phone. The primary goal of the thesis is to learn about programming and how to use Unity.

The idea of the game is to move a ball through a labyrinth while gathering points and avoiding traps. The game can be won by reaching a flag. The main targets for the game are children and also older casual players. The platform is Android-devices, although I have tested this a lot with PC.

Whole game is made with Unity 3D-game engine and the scripting language is C#. Most of the objects in the game have been downloaded from the Unity Asset Store and Google have been used a lot for getting textures and sound effects. Also many tutorial-videos have been watched from Unitys page and Youtube. Other programs that have been used are Microsoft Visual Studio for editing scripts and Blender for 3D-modeling.

The basic idea succeeded very well, although to commercialize it is still kinda far and some of the more sophisticated features require bit more adjustment. The final game is going to be released at Google Play Store.

Keywords Unity 3D, C#, Android, velocitysensor, game

# SISÄLLYS

# 1 JOHDANTO

Minulla on ollut mielenkiintoa pelien tekemiseen jo pienestä pitäen. Amiga 500-koneellani tein MikroBitissä olleeseen laskukoneen koodiin perustuneen pelin, joka oli todella huono. Minulla oli myös 3D Construction Kit II, joka oli kuitenkin koneelleni liian raskas. Varhaiset kokemukset musiikin tekemisestä ovat myös tuolta ajalta. Deluxe Paintilla piirtelin sprite-grafiikoita ja tein myös animaatioita. Idea pelin tekemisestä päättötyöksi tuli toisena opiskeluvuonna. Silloinen luokanvalvojani Pirjo Prosi sanoi, että peli päättötyönä kelpaa hyvin, mutta alustan olisi hyvä olla joku muu kuin PC.

Muistin isoäidilläni olleen puusta tehdyn mekaanisen Labyrintti-pelin ja ajattelin, että tämänhän voisi toteuttaa puhelimelle ohjauksen toimiessa puhelinta kallistamalla. Tällöin ei ollut vielä mitään tietoa kiihtyvyysantureista, C#:sta eikä Unity 3D:stä. Pirjo opetti meille silloin JavaScriptiä, mikä oli ensimmäinen ohjelmointikieli, josta tykkäsin. Siksi tässä on hiukan onnea mukana, koska Unity 3D ohjelmoidaan C#-kielellä, mikä on hyvin lähellä tuota JavaScriptiä.

Halusin nimenomaan tehdä jotain itselleni, enkä ulkopuoliselle organisaatiolle vaikka siitä maksettaisiin. Tämän takia toimeksiantajani on käytännössä VAMK, vaikka itse idea olikin minun. Uskon, että tästä pelistäni on hyötyä VAMKille tai jollekin koulun oppilaalle. Kun tieto pelistäni (joskus toivottavasti) leviää laajemmalle, saattaa se inspiroida jotain hakemaan VAMKiin, ”Täällä oppii tekemään myös pelejä.”. Joku uusi tai nykyinen oppilas voi innostua pelin tekemisestä ja sitä kautta myös ohjelmoinnista. Pelini on kuitenkin näennäisestä yksinkertaisuudestaan huolimatta kohtalaisen näyttävä.

Kyseessä on siis yksinkertainen pallo labyrintissä-peli mobiilialustalle (Android-laitteet), jossa ohjaus perustuu laitteen kallistamiseen. Alkuperäisessä pelissä lattiassa on reikiä, joihin ei saa tipahtaa. Tietokone kuitenkin mahdollistaa kaikenlaisia asioita, mitkä eivät ole tuollaisessa mekaanisessa mahdollisia. Kehittyneemmässä versiossa on korvattu reikiä erilaisilla ansoilla, lisätty ovia, jotka saattavat

avautuakseen vaatia avaimen ja lisätty myös muunlaista visuaalista ilmettä. Pallo esimerkiksi räjähtää osuessaan ansaan. Lisäksi pelissä on ääni ja musiikki.

Kenttätyyppejä on kolmenlaisia, perinteiset kentät, kehitetyn sisällön kentät ja ohjelmointiaiheiset kentät. Peli sisältää menun, muutaman kentän, creditsit sekä voitto- ja kuolemaruudut. Perusidea toimii, mutta tässä on vielä paljon tekemistä, että olisi julkaisukelpoinen Google Play Storessa. Maksullisen sisällön toteuttaminen vaatii lisäksi aika paljon perehtymistä vielä.

Alkuun ajatteltu tehdä lisää pelkästään tuollaisia kehitetyn sisällön kenttiä, mutta testaajilta palautetta saatua kävi ilmi, että naispuoliset henkilöt pelaisivat mieluummin itse tuota peruspeliä. Tuollaisten kenttien tuottaminen on loppujen lopuksi helppoa. Internetissä on sivustoja, jotka tuottavat satunnaisia sokkeloita. Toki näiden Internetissä luotujen pohjapiirrosten muuttaminen pelikentiksi vaatii puurtamista.

Vaativimpien kenttien kehittämiseen on ajateltu käyttää Axis Game Factoryä, mikä on maksullinen editori Unitylle. Tällä pystyy tekemään jo kohtalaisen hyvää grafiikkaa ja jos niitä ei käytetä tässä pelissä, voi niitä käyttää mahdollisesti jossain muussa pelissä. Esittelyversiossa ei ollut mitään tällä tehtyä ja ollaan vasta tutkimassa miten se toimii. Axis Game Factoryllä kuitenkin pystyy piirtämään erikorkuista maastoa, lisäämään puita, vettä, käytäviä, linnan torneja, tuulimyllyjä ja vaikka mitä.

On myös ostettu maksullisia 3D-objekteja, joilla saa hiottua tuota perusgrafiikkaakin hiukan paremmaksi. On silti ajatellut tehdä jotain 3D-objekteja itsekin. Näitä kun sitten yhdistelee, niin hyvä tulee. Johtopäätöksenä voisi sanoa, että Unityllä pelien tekeminen on helppoa, kun on ohjelmoimisen perusteet hallussa ja mielenkiintoa riittää. Jos taas pitäisi tehdä pelimoottori alusta asti itse, tuskin saisin kovin ihmeellistä aikaiseksi.

Minua on kuitenkin auttanut tässä koulutuksen lisäksi pitkäaikainen kiinnostus, kuvankäsittelyohjelmien osaaminen ja ystäväni, jotka tekevät musiikkia. Kaksi

heistä tekee periaatteessa tilauksesta lisää, jos kertoo vähän mitä tyyliä haluaa. Itse tekemääni musiikkia en laittanut tähän, koska ne eivät ole mielestäni tarpeeksi hyviä. Olen lisäksi pelannut niin monimutkaisia pelejä (rooli- ja strategiapelit, sekä lentosimulaattorit), että tiedän paljon pelimekaniikasta mikä ei kuitenkaan vielä tässä projektissa sinänsä paljoa auttanut, koska pelityyppi on aika yksinkertainen. Varsinkin kynällä, paperilla ja nopilla pelattujen roolipelien tuntemus auttaa tietämään mitä tapahtuu, miten tapahtuu ja miksi tapahtuu.

# 2 KÄYTETTYJÄ KÄSITTEITÄ

**HUD:** Heads-up-display. Tämä suihkuhävittäjistä tuttu ominaisuus tulostaa pelimaailman päälle grafiikkaa ja tekstiä. Pitää siis kirjaa pelaajan scoresta, elämistä, kerätyistä avaimista sekä tulostaa ruudulle viestejä, jos ne on triggeröity pelimaailmassa. Ohjelmointiaiheisissa kentissä on hiukan riisutumpi versio tästä.

**Objekti:** Pelissä oleva esine, esim. pallo, ansa, kamera tai muu. Näihin on monesti lisätty joku skripti, mutta voivat olla myös staattisia, kuten seinät.

**Skripti:** Koodinpätkä, poikkeuksetta C# kielellä.

**Skene:** Nämä ovat erilaisia kohtauksia pelissä, esim. menu-, kuolema- ja voittoruudut, mutta myös eri kentät ovat scenejä.

**Triggeri:** Tämä laukaisee jonkun tapahtuman pelissä, yleensä pallon osuessa johonkin tapahtuu jotain, riippuen objektissa olevasta tägistä.

**Tägi:** Pelissä olevia objekteja voi tägätä, esim. Player tai Hazard. Tätä tägiä sitten verrataan onko sama kuin koodissa vaadittu tägi.

**Unity 3D:** Ilmainen pelimoottori ja editori, jolla peli on toteutettu. Sisältää myös maksullisia ominaisuuksia, mitkä eivät ole kuitenkaan tässä käytössä.

**Unity Asset Store:** Paikka, josta saa maksullisia ja ilmaisia peliobjekteja ja muuta tavaraa. Tällä hetkellä kaikki tuolta ladatut ovat ilmaisia.

# 3 PELI

## 3.1 Pelin tavoite

Pelissä ohjataan palloa Android-laitetta kallistamalla. Kuten pelin nimikin sanoo, kenttien on tarkoitus olla labyrinttimaisia. Kentän läpäisee pääsemällä lipun luo. Kentissä on kerättäviä esineitä, kuten kolikoita ja avaimia. Kolikoista saa pisteitä, joiden määrä riippuu kolikon arvosta. Avaimilla voi avata lukittuja ovia. Jotta peli ei olisi liian helppo, on siellä tietenkin ansoja. Ansaan osuessa menettää elämän, joita on alkuun viisi, jotka menetettyään joutuu aloittamaan pelin alusta (kuolee). Ohjelmointiaiheisista kentistä on jätetty pois tuo kuolemaominaisuus ja kerättävät esineet, kentän pääsee läpi ratkaisemalla puzzlen.

## 3.2 Pelaajahahmo

Pelaajahahmo pelissä on siis pallo ja se on nimetty Amigapalloksi tuon puna-valkoruutuisen tekstuurin mukaan. Amigapallo on esiintynyt ensimmäisen kerran 1984 julkaistussa demossa, missä esitellään Amiga 500-tietokoneen suoritustehoa. Tämä pallo on hyvin tunnettu tietyissä piireissä ja on saatu sen tunnistaneilta positiivista viestiä asiasta. Projektin PC-versiossa oleva hiiren kursori on myös Amigan kursori. Pallo on luotu yksinkertaisesti luomalla sphere-objekti ja lisäämällä siihen tekstuuri, räjähdysääni, räjähdysanimaatio, kaksi koodia ja muutama fysiikkaominaisuus.

## 3.3 Pelissä olevat skenet

Tähän alle on listattu pelissä käytettyjä scenejä ja muutama suunnitteilla oleva.

**Main Menu**

Kun pelin käynnistää, Main Menu (**Kuva 1.**) on ensimmäinen scene joka aukeaa. Se sisältää pelin logon, eri kenttien käynnistysnappulat ja pari mainosta.

Kuva 1. Main Menu

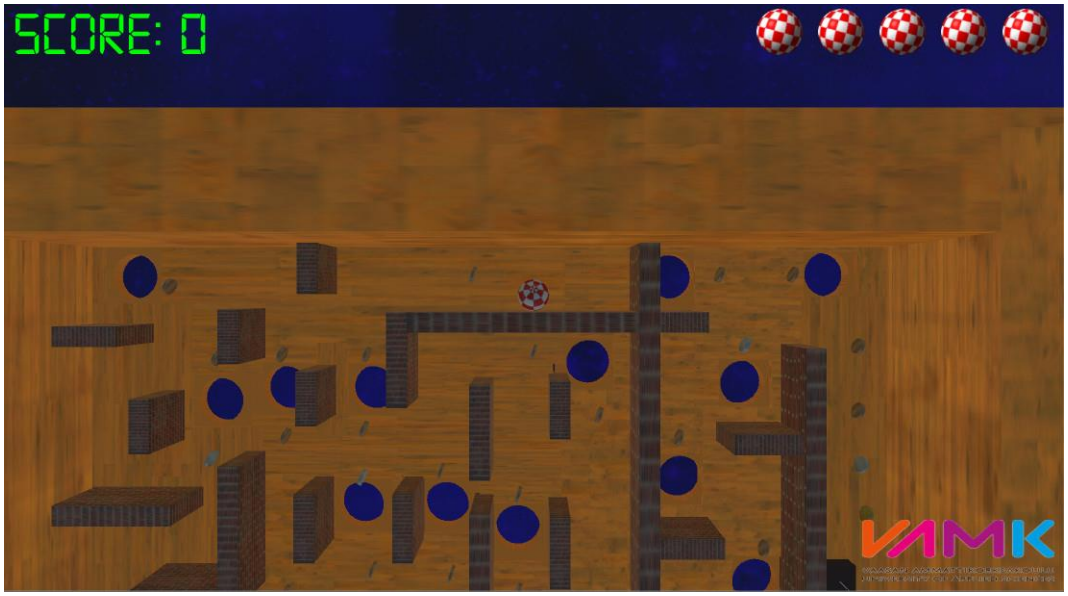
**Tutorial – Harjoitusmoodi**

Harjoitusmoodissa neuvotaan miten peliä pelataan. Moodi ei ehtinyt tähän versioon.

- Miten palloa liikutetaan.
- Mitä kannattaa kerätä (kolikot, timantit, sydämet, avaimet).
- Mihin ei kannata osua (ansat).
- Miten avaat lukitun oven.
- Miten sammutat generaattorin.
- Missä on maali.

**Original Level**

Original level (**Kuva 2.**) on alkuperäinen Labyrintti kopioituna lähes 1:1 perusidean testaamista varten. Tätä ei tule kuitenkaan lopulliseen versioon, vaan se on luotu perusidean testaamista varten.



Kuva 2. Original level

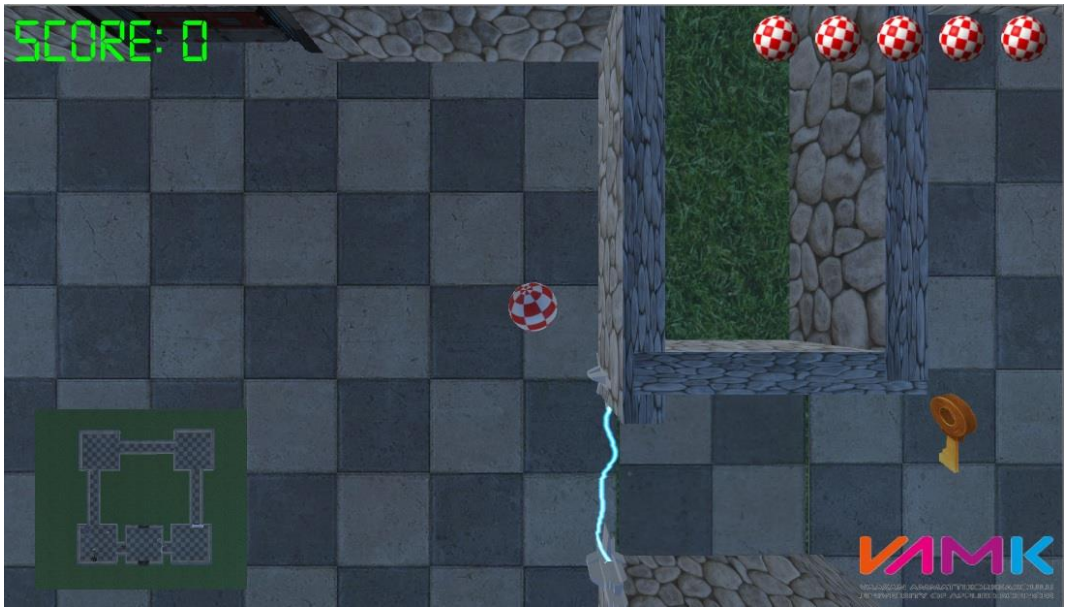
**Constructor Level**

Constructor Level (**Kuva 3.**) on nimetty hiukan harhaanjohtavasti, eli kyseessä ei ole kenttä jossa pelaajat saisivat tehdä omia kenttiään. Tuo scene on pohja, mistä lähdetään monistamaan uusia kenttiä ja nimi jäi tuollaiseksi tähän demoversioon. Tästä saa kuitenkin hyvän käsityksen siitä, mitä uutta sisältöä on luotu, jota esikuvassa ei ole. Tietokonehan mahdollistaa kaikenlaisia ominaisuuksia, mitä ei fyysisesti toteuteta ainakaan kovin kilpailukykyiseen hintaan. Lisäksi tuollainen pyörivä sirkkelinterä ei varmaan ole kovin turvallinen, jos joku lapsi pelaa.

Kuva 3. Constructor Level

**Castle Level**

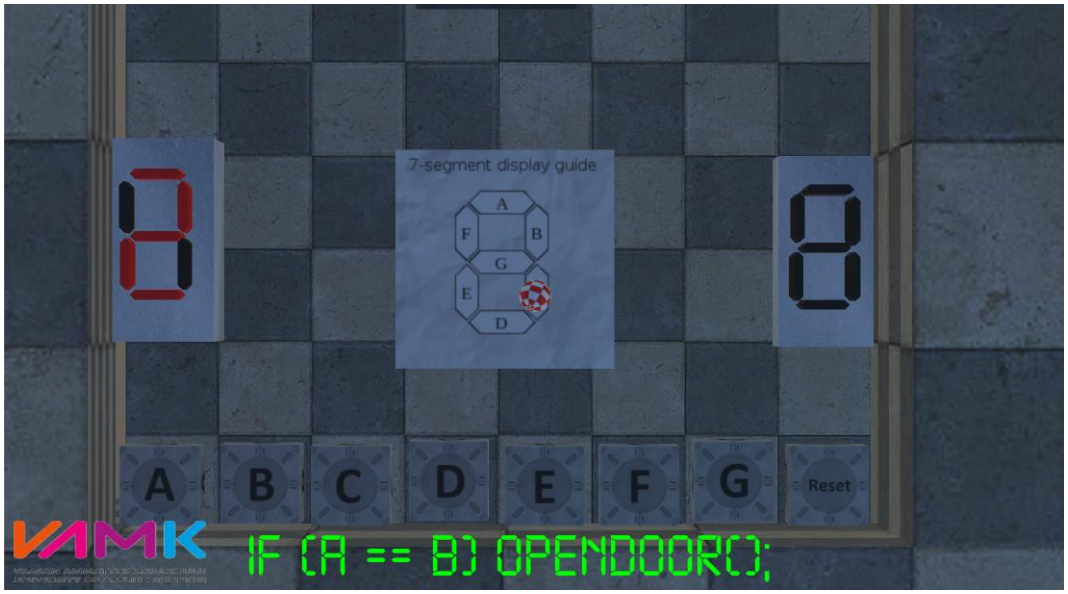
Castle Level (**Kuva 4.**) on vasta suunnitteluasteella, esittelee kuitenkin sitä, että kentät voivat olla huomattavasti isompiakin kuin edelliset. Kun tuo alkuperäinen mahtuu hyvin ainakin tabletilla yhdelle ruudulle, tämä on jo niin iso, että siihen on lisätty minikartta vasempaan alakulmaan suunnistamista helpottamaan.



Kuva 4. Castle Level

**7-Segment Display**

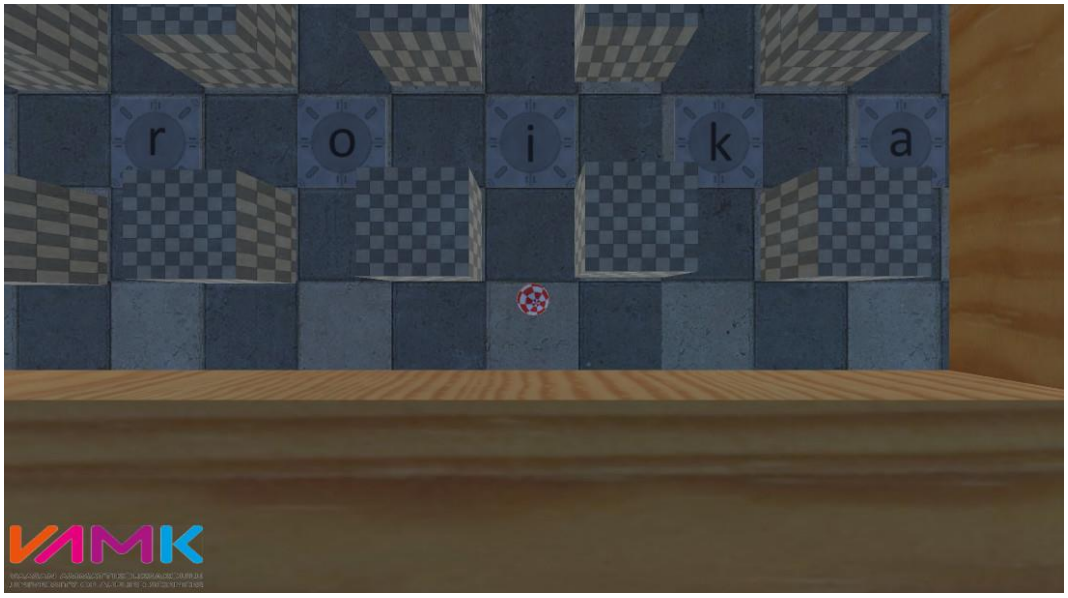
7-Segment Display on ensimmäinen ohjelmointiaiheinen kenttä. Tässä käsitellään if-lauseketta ja seitsemän-segmenttinäytön ohjelmointia. Ensimmäisessä scenessä tuo if ja seitsemän-segmenttinäyttö (**Kuva 5.** 7-Segment Level). Aluksi ensimäinen näyttö arpoo satunnaisen numeron ja näyttää sen. Toinen näyttö ei näytä mitään, mutta kirjaintunnuksellisia painolaattoja painamalla saa segmenttejä syttymään. Jos sytytti vahingossa väärän ”LEDin”, pitää se resetoida. Kun numerot molemmilla näytöillä ovat samat, ovi aukeaa ja pääsee toiseen huoneeseen, jossa maali odottaa.



Kuva 5. 7-Segment Level

**Infinite Loop**

Toinen ohjelmointiaiheinen kenttä on Infinite Loop (**Kuva 6.**). Tässä on eräänlainen matriisi, jossa on painolaattoja kirjaimilla. Huoneen toisella puolella häämöttää maali, jolla kenttä läpäistään ja sinne voi rynnätä välittämättä painonapeista. Maaliin tullessa peli alkaa kuitenkin valittamaan ikuisesta loopista ja pallo siirtyy takaisin alkuun. Avain kentän läpäisyyn on siinä, että painolaatat aktivoidaan oikeassa järjestyksessä, eli muodostetaan sana ”break;”.



Kuva 6. Infinite Loop

Näitä ohjelmointiaiheisia puzzleja lisätään myöhemmin. Puzzlejen toteuttaminen on hyvää harjoitusta loogisen ajattelun kehittämiseksi.

**Shop – Kauppa**

Kaupassa voi ostaa oikealla rahalla pelin sisäistä valuuttaa, eli timantteja. Tämä ei ehtinyt tähän versioon, mutta alla suunnitelma miten se toimisi.

- Kymmenen timanttia maksaa 1E.
- Kymmenellä timantilla aukaisee viisi maksullista kenttää.
- Yhdellä timantilla ostaa yhden lisäelämän tai panssarin.
- Huom! Timantteja voi kerätä myös itse pelissä, mutta ne ovat harvinaisia.

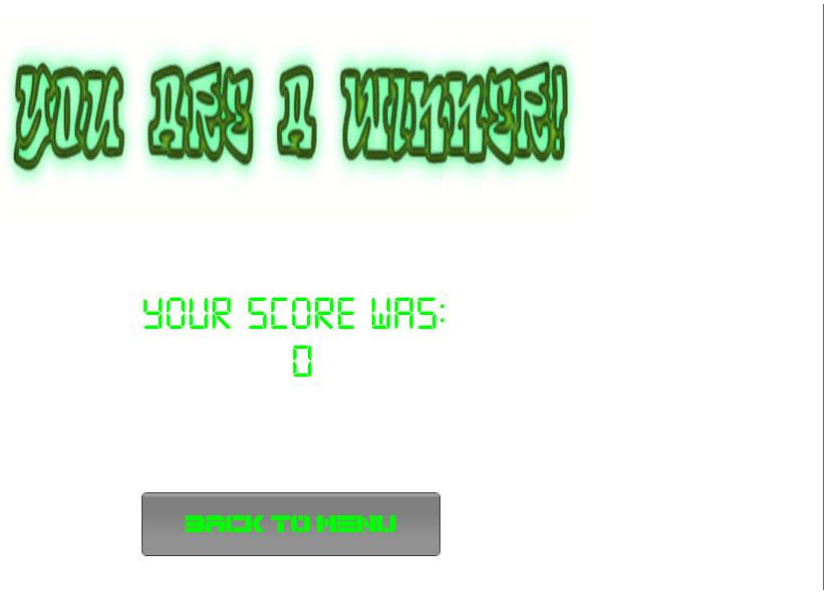
### Credits

Creditsit, eli lopputekstit (**Kuva 7.** Credits), kertovat tekemiseen tekijän, henkilöt jotka ovat auttaneet minua projektissa ja kiitokset. Tämä on tälläinen demomainen: musiikki soi, pallot pomppii ja teksti vierii ruudun laidalta laidalle.

Kuva 7. Credits

### Win - Voitto

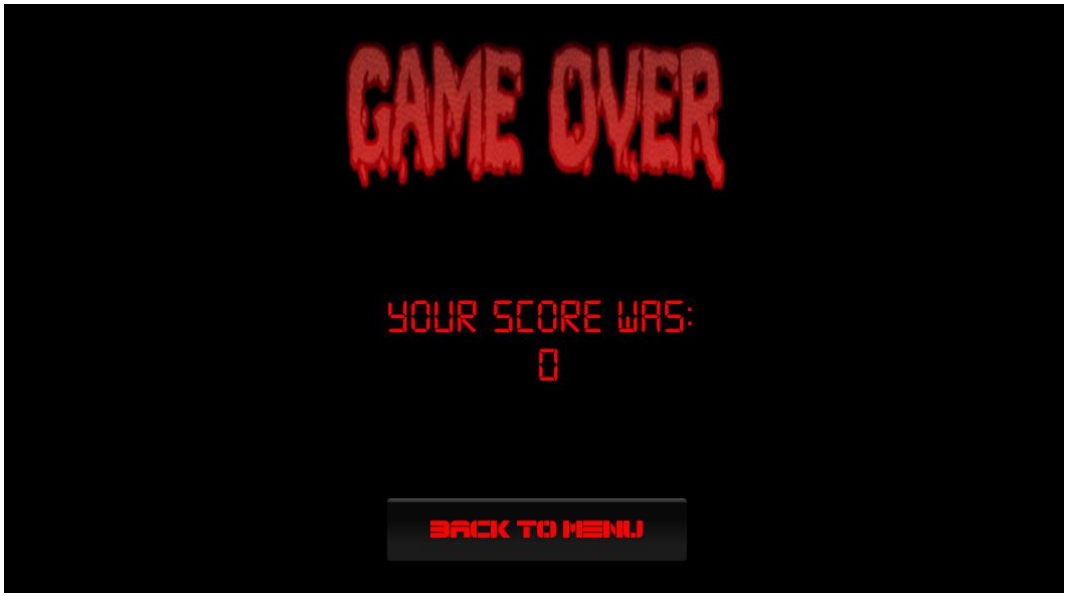
Win-ruutuun (**Kuva 8.)** päätyy kun läpäisee kentän. Lopullista scorea ei ehditty saamaan toimimaan. Lopullisen version pitäisi sisältää myös highscore-listan.




Kuva 8. Win scene

**Death - Kuolema**

Death ruutuun (**Kuva 9.**) päätyy kun kuolee. Muuten pitäisi olla sama kuin voitto-ruutu.




Kuva 9: Death scene

## 3.4 Pelissä olevat kerättävät esineet

Pelissä on siis objekteja joita voi kerätä, alla on lista niistä. Unityssä tägätty aina Collectibleksi, vaikka se ei ole sinänsä tarpeen koodin puolesta.

**Kuparikolikko**

Yhden pisteen arvoinen, lisää pisteen pelaajan tulokseen (Score).

**Hopeakolikko**

Viiden pisteen arvoinen, lisää pisteet pelaajan tulokseen (Score).

**Kultakolikko**

Kymmenen pisteen arvoinen, lisää pisteen pelaajan tulokseen (Score).

**Kulta-avain**

Kun pelaajalla on tämä, ovi joka vaatii sen, aukeaa.

**Hopea-avain**

Lähes sama kuin kulta-avain, eli avaa lukitun oven.

Koska ei ole ehditty toteuttamaan kaikkia suunniteltuja ominaisuuksia, tässä alla on objekteja, jotka eivät ehtineet päättötyön esittelyyn. Peliobjektit näille on it-seasiassa olemassa, mutta koodit niille puuttuvat. Sinänsä ei ole välttämättä kovin vaikeita koodeja, tuota timanttia lukuunottamatta, koska ei ole ehditty perehtymään tuohon kauppaominaisuuteen vielä ollenkaan.

**Timantti**

Tuhannen pisteen arvoinen. Kymmenellä timantilla saa avattua uuden kentän. Timantteja voi ostaa myös oikealla rahalla, jolloin yhdellä eurolla saa kymmenen timanttia. Tarkoitus sijoittaa vaikeisiin paikkoihin.

**Sydän**

Lisäelämä. Näitä sellaisiin paikkoihin, että näitä tavoitellessa on suuri vaara menettää elämän.

**Panssari**

Suojaa palloa elämän menetykseltä yhden kerran. Ei kuitenkaan auta, jos pallo putoaa reiästä. Lisää pallon painoa hiukan, jolloin ohjaus on hiukan erilainen. Myös pallon tekstuuri vaihtuisi Amiga-tekstuurista teräksisen näköiseksi.

## 3.5 Pelissä olevat vaarat

Pelissä on vaaroja, jotka aiheuttavat elämän menetyksen. On tägätty nämä kaikki Hazardeiksi ja tämä on tärkeää, koska elämän menetyksen toteuttava koodi tutkii törmättiinkö Hazardiin.

**Reikä**

Lattiassa on reikä, josta pudotessa menettää elämän. Jos olisin lisännyt panssarin peliin, ei se kuitenkaan suojaisi tältä. Olisi helppo tehdä näitä myös siten, että joku kytkin sulkisi ja aukaisisi niitä.

**Sirkkeli**

Pyörivään sirkkeliin osuttaessa menettää elämän. Jos pelissä olisi panssari, suojaisi se törmätessä yhden kerran. Pysähtyneeseen sirkkeliin osuttaessa ei tapahdu mitään. Sirkkeleitä voisi myös pysäyttää tai käynnistää kytkimillä tai jos sammuttaa generaattorin, kaikki sirkkelit pysähtyvät.

**Sähköseinä**

Sähköseinään törmättäessä menettää panssarin tai elämän. Näitäkin voi sulkea ja avata kytkimillä tai sammuttamalla generaattorin.

**Piikkiansa**

Piikit voivat olla joko ylhäällä tai alhaalla. Nousevat tai laskevat aina aikaviiveen jälkeen. Jos piikit ovat alhaalla, ei niistä tapahdu mitään, kun taas ylhäällä olevista piikeistä menettää panssarin tai elämän. Piikkiansoja voi kerätä ryhmiksi, jolloin ne nousevat ja laskevat sekvensseissä. Piikkiansasta pääsee läpi oikealla ajoituksella.

**Monsters - Hirviöt**

Hirviöt ovat käytännössä samanlaisia kuin vaarat sillä erotuksella, että ne liikkuvat kartalla joko ennalta määrättyä reittiä, tai noudattavat eräänlaista tekoälyä esim. seuraavat pelaajaa. Hirviöt on tägätty myös hazardeiksi.

## 3.6 Muut peliobjektit

Aktivaattorit ovat periaatteessa kytkimiä, joihin osuttaessa tapahtuu jotain. Käytännössä muuttaa nollan ykköseksi samassa tai jossain toisessa scriptissä, jolloin kyseinen scripti tekee jotain.

**Painolaatta**

Sulkee tai aukaisee yhden tai useamman oven, ansan tai jonkun muun objektin. Näitä on paljon noissa ohjelmointiaiheisissa kentissä ja niiden toimintalogiikka on hiukan monimutkaisempi kuin peruspelissä olevissa.

**Viesti-triggeri**

Pelissä on objekteja, joihin osuminen aktivoi viestin ruudulle. Käytännössä tämä on toteutettu lisäämällä kenttään kuutio, josta poistettu renderer ja lisätty koodi. Muuttaa siis HUD-skriptissä viestin tilaksi ykkösen, jolloin viesti näytetään.

**Ovi**

Jos kyseessä on tavallinen ovi (eli ei painolaatalla tai avaimella aukeava), aukeaa se siihen törmäämällä. Toisenlaiset ovet aukeavat joko painolaatan avulla tai oikean avaimen hankkimisella.

**Generaattori**

Sulkee tai aukaisee kaikki sähköä vaativat objektit, kuten sirkkelin tai sähköseinän. Ei löydetty mistään ilmaiseksi animoitua höyrykonetta, eikä ole ehditty miettimään miten moinen toteutettaisiin, mutta tämä täytyy kyllä lisätä lopulliseen peliin, koska se olisi aika näyttävä.

**Hyppyri**

Joidenkin ansojen yli voi hypätä myös, jos vauhtia on tarpeeksi. Ei ole toteutettuna vielä tässä versiossa. 3D-malli tälle ominaisuudelle on helppo toteuttaa.

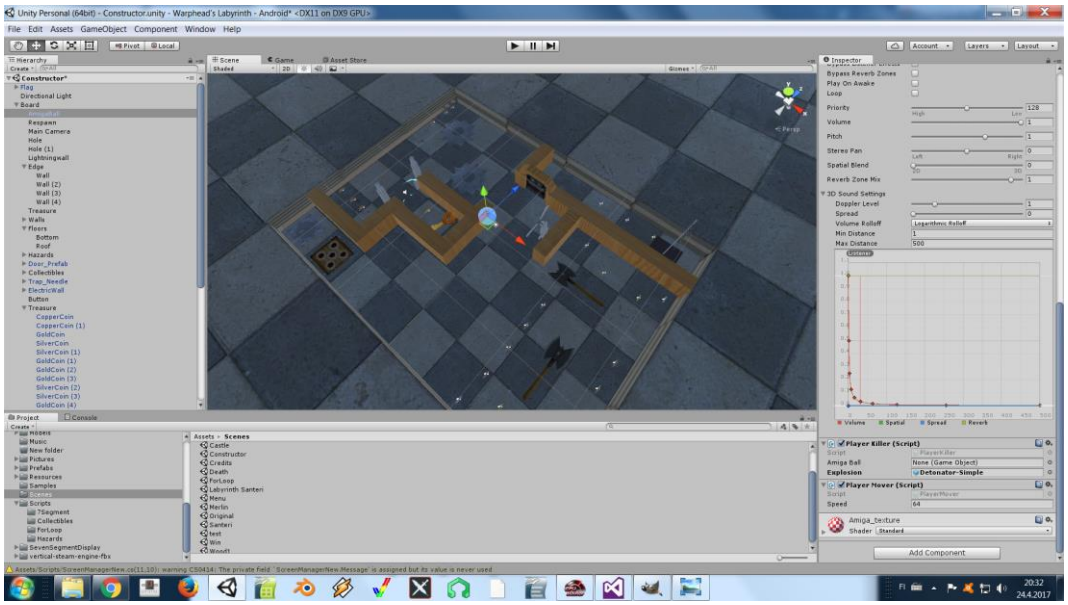
**Seitsemän-segmenttinäyttö**

Pelissä on kaksi erilaista seitsemän-segmenttinäyttöä. Toinen arpoo itseensä numeron ja näyttää sen. Toiseen saadaan numero aktivoimalla inputteina olevia painolaattoja. Sisältää myös resetin. Unity Asset Storessa oli ilmainen seitsemän-segmenttinäyttö, joka oli kuitenkin ohjelmoitu jollain toisella kielellä kuin C#, joten en ymmärtänyt siitä paljoa. Siksi ajattelin, että kun saan noiden näyttöjen ulkoasua hiukan paremmaksi, ajattelin julkaista ne tuolla Asset Storessa ilmaiseksi. Tällä hetkellä nuo näyttöjen ”LEDit” ovat ainoat 3D-objektit, jotka on tehty itse.

# 4 EDITORIT

## 4.1 UNITY 3D

Tässä käsitellään lyhyesti hiukan Unity3D-editoria. Kovin syvällisesti asiaan ei puututa. Ensiksi yleisnäkymä (**Kuva 10**):



Kuva 10. Unity 3D

Vasemmalla yläpuolella on skenen sisältö. Kuten näkyy, niitä kertyy helposti paljon, joten niille kannattaa tehdä kansioita (luodaan tekemällä tyhjä peliobjekti). Heti sen alla on projektikansion sisältö, tiedostot erikseen tuossa keskellä. Keskellä näkymä skenestä, samaan ruutuun aukeaa myös pelinäkymä tai Asset Store. Oikealla on valitun peliobjektin (tällä hetkellä pallo) sisältämät asiat.

Se sisältää tiedon objektin nimestä ja kertoo, että sen tägi on ”Player”. Tämä tägi on oleellinen useammankin koodinpätkän takia. Transform-kohdassa on sen sijainti pelimaailmassa, mihin suuntaan se katsoo ja sen koko. Sphere collider huolehtii törmäyksien tarkistamisesta. Mesh renderer liittyy pallon mallintamiseen ja siihen liitettyyn tekstuuriin.

Rigidbody on fysiikkaominaisuus, joka liittyy pallon liikkumiseen. Pallolla on massa ja Unityssä on mahdollista myös mallintaa tuulenvastusta painovoiman lisäksi. Constant force saattaa minulla olla turhaan tässä, koska pallolle on määritelty jo tuo massa, joka on tosin varsin pieni tuolla Y-akselin -111 on kuitenkin mallinnettu tuota painovoimaa hiukan eri tavalla.

Audio source on palloon liitetty wave-tiedosto, jossa ei sen kummempia säätöjä ole määritelty. Unity mahdollistaa kuitenkin äänenvoimakkuuden etäisyyden mukaan ja erilaisia efektejä, esimerkiksi Reverb muuttaa ääntä siten, että se vaikuttaa kuuluvan jossain isossa hallissa, missä ääni kaikuu seinistä.

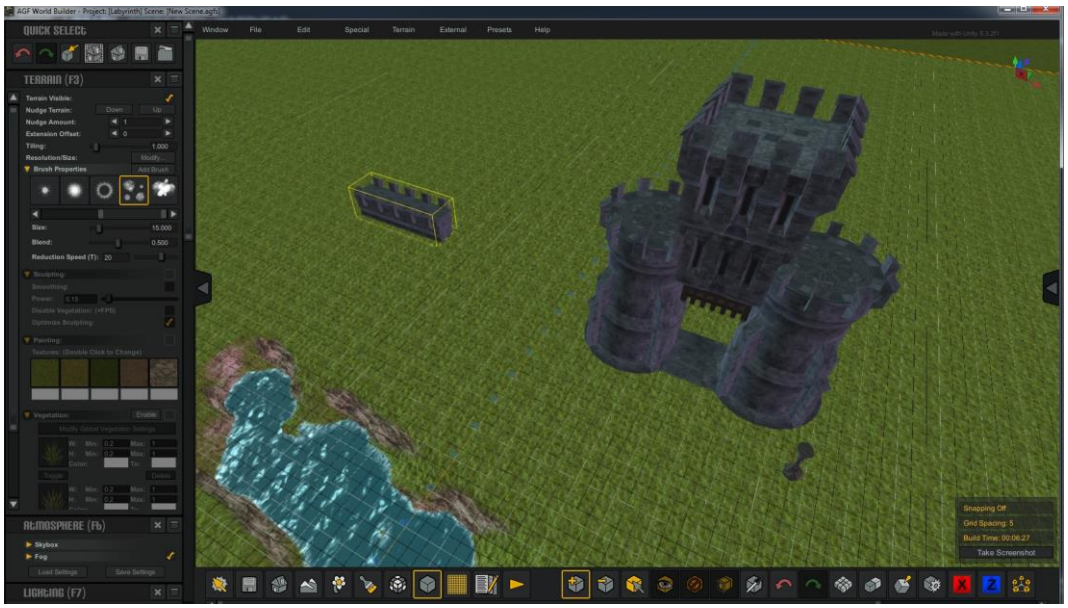
AmigaBall-peliobjektiin on liitetty Player Killer-skripti, joka huolehtii elämien menettämisestä ja peliobjektiin on liitetty myös Detonator-simple räjähdys-animaatioksi. Player Moverissa on pallon liikuttamiseen tarvittava koodi, sisältää ohjauksen PC:llä ja Androidilla.

## 4.2 MICROSOFT VISUAL STUDIO

Visual Studiota on käytetty skriptien kirjoittamiseen, sinällään aika tyypillinen editori. Sisältää kuitenkin Unity-tuen, ja osaa kertoa jos syntaksi on väärä. Vaatii JREn toimiakseen.

## 4.3 AXIS GAME FACTORY

Axis Game Factory (**Kuva 11.**) on maksullinen editori, ei ole ehditty juurikaan paljon käyttämään, alla kuitenkin minuutissa tehty kuva demoksi. Axis Game Factoryllä pystyisi tekemään jonkinlaisia pelejä itselläänkin ja se sisältää yllättäen ominaisuuden, jolla kenttiä voi testata first-person- tai third-person-perspektiivistä. Tätä ei kuitenkaan aiota käyttää muuten kuin kenttien tuottamiseen ja koostetaan itse pelit Unity 3Dssä.




Kuva 11. Axis Game Factory

# 5 VAATIMUSMÄÄRITTELY

**Mitä ja kenelle?**

Android-peli, joka on tarkoitettu lapsille ja myös vanhemmille, kuitenkin lähinnä satunnaisille pelaajille. Pelissä on sokkelomaisia kenttiä ja ansoja, joihin ei saa osua. Pelissä pitää olla myös ohjelmointiaiheisia puzzleja, esim. matriisi, josta pääsee läpi muodostamalla ”BREAK;”. Pitää olla myös kerättäviä esineitä, joita keräämällä saa pisteitä. Kentät läpäistään pääsemällä maaliin, eli lipun luo. Lähden kehittämään peliä mahdollisesti kolmeen eri suuntaan, hyvin pelkistettyyn versioon ja vähän monimutkaisempaan, jossa sisältöä on kehitelty. Kolmannessa versiossa yhdistetään ohjelmointiaiheiset kentät samaan skeneen ja korvataan pallo robotilla (käy hyvästä harjoituksesta). Tämä ei tule sisältämään itse peruspeliä ja sen tarkoitus on toimia mainoksena VAMKille. Muut versiot sisältävät myös VAMK-logon. Peli pitää julkaista Googlen Play Storessa ja siihen täytyy kehittää pelin sisäisiä ostoja. Erilaisia kenttiä täytyy olla lopulta useita kymmeniä.

| V# | Vaatimus | Prioriteetti |
|---|---|---|
| V1 | Pelialusta on Android | 1 |
| V2 | Ohjaus perustuu laitteen kallistamiseen | 1 |
| V3 | Pelimoottori on Unity 3D | 1 |
| V4 | Pelihahmo on pallo | 2 |
| V5 | VAMK-versiossa pelihahmo robotti | 3 |
| V6 | Pelissä on vaaroja, joihin osumalla menettää elämän | 2 |
| V7 | Pelissä on kerättäviä esineitä, joista saa pisteitä | 2 |

| | | |
|---|---|---|
| V8 | VAMK-versiossa on ohjelmointiaiheisia puzzleja | 3 |
| V9 | Lopullinen peli julkaistaan Google Play Storessa, mahdollisesti kahtena eri versiona. | 1 |
| V10 | Peli tulostaa tietoa ruudulle, kuten elämien määrän, pistemäärän, kerätyt esineet ja viestejä. | 2 |
| V11 | Pelissä on ääni ja musiikki | 2 |
| V12 | Peli sisältää sponsoreiden ja VAMKin mainokset | 3 |
| V13 | Pelissä on kenttiä | 2 |
| V14 | Pelin voi pelata läpi | 3 |

# 6 TOIMINNALLINEN MÄÄRITTELY

## 6.1 Johdanto

Tarvitttu aihe päättötyölle ja on jo pitkään haluttu tehdä pelin Alunperin peli oli tarkoitettu joko myytäväksi Googlen Playstoressa tai jaettavaksi ilmaiseksi, mahdollisesti saaden tuloja mainoksista tai ns. in-game purchaseista. Ensimmäisessä päättötyön ohjauksessa kuitenkin sovittiin, että peli tehtäisiin mainokseksi VAM-Kille. Peli on tarkoitettu nuorille ja miksei vanhemmillekkin satunnaispelaajille. Pelimoottoriksi valitsin Unity 3D:n ja skriptaamisen teen pääasiallisesti C#:lla.

## 6.2 Yleiskuvaus

Käyttäjä on pelaaja, käyttöympäristö Android, liittymät ympäristöön ovat kosketusnäyttö ja kiihtyvyysanturit.

## 6.3 Tiedot ja tietokanta

Pelissä ei ole muita tietokantoja kuin hiscore-lista (eli parhaimmat tulokset). Ainoa tallennettava tieto on oikeastaan se, että haluaako pelaaja äänten ja musiikin olevan päällä vai poissa.

## 6.4 Käyttötapausten läpikäynti

### 6.4.1 Lataa kenttä

| ID | Nimi |
|---|---|
| Kuvaus | Lataa kenttä |
| Viittaus vaatimukseen | V13 |
| Esiehdot | Käyttäjä on menussa ja painaa nappia |
| Käyttäjä | Normaali käyttäjä |
| Normaali toiminta | 1. Lataa halutun kentän |
| Poikkeustilanteet | Jos painetaan ”Quit”, sovellus lopetetaan |
| Tulos | Kenttä latautuu |
| Toteutussuunnitelma | Kentän nimi on nappulassa, joka on linkitetty koodiin ja joka lataa kentän |

### 6.4.2 Pallon liikuttaminen

| ID | Nimi |
|---|---|
| Kuvaus | Pallon liikuttaminen kiihtyvyysanturilla |
| Viittaus vaatimukseen | V2 ja V4 |

| | |
|---|---|
| Esiehdot | Käyttäjä on itse pelissä |
| Käyttäjä | Normaali käyttäjä |
| Normaali toiminta | 1. Pallo liikkuu laitteen kallistuksen mukaan |
| Poikkeustilanteet | Pallo osuu seinään tai ansaan |
| Tulos | Pallo liikkuu |
| Toteutussuunnitelma | Sovellus lukee laitteen kiihtyvyysanturia, joka koodilla muutetaan pallon liikkeeksi. |

### 6.4.3 Esineiden kerääminen

| ID | Nimi |
|---|---|
| Kuvaus | Erilaisten pelissä olevien kerättävien esineiden hankinta |
| Viittaus vaatimukseen | V7 |
| Esiehdot | Käyttäjä on pelissä ja pallo osuu kerättävään esineeseen. |
| Käyttäjä | Normaali käyttäjä |
| Normaali toiminta | 1. Lisää pelaajalle pisteitä riippuen kerättävälle esineelle määritellystä pistearvosta. |

| | |
|---|---|
| Poikkeustilanteet | Jos kyseessä on avain ei lisätä pisteitä, vaan lisätään avain inventaarioon. |
| Tulos | Pisteet lisääntyvät |
| Toteutussuunnitelma | Kerättävissä esineissä on koodinpätkä, mikä jonkun objektin törmätessä siihen, tutkii onko kyseessä pelaaja. |

### 6.4.4 Voittaminen

| ID | Nimi |
|---|---|
| Kuvaus | Kentän voittaminen |
| Viittaus vaatimukseen | V14 |
| Esiehdot | Käyttäjä on läpäissyt kentän ja päässyt lipulle asti. |
| Käyttäjä | Normaali käyttäjä |
| Normaali toiminta | 1. Pelaajan tulos tallentuu.<br>2. Voitto-scene latautuu.<br>3. Ruudulle tulostetaan teksti missä ilmoitetaan voitosta.<br>4. Pelaajan tulos tulostuu.<br>5. Jos pelaajan tulos suurempi kuin edellinen |

| | |
|---|---|
| | highscore, tallennetaan se.<br><br>6. Highscore-lista tulostuu. |
| Poikkeustilanteet | - |
| Tulos | Voitto-scene latautuu ja hiscore lista näkyy. |
| Toteutussuunnitelma | Lippuun on linkitetty koodi, joka tutkii onko siihen osuneen peliobjektin tägi player, jos on niin peli on voitettu. Voitto-scenessä on koodi joka huolehtii hiscoren ja muun tekstin näyttämisestä. Sama koodi vertaa scorea edellisiin hiscore-listalla oleviin numeroihin. |

### 6.4.5 Kuoleminen

| ID | Nimi |
|---|---|
| Kuvaus | Kuoleminen. |
| Viittaus vaatimukseen | V6 |
| Esiehdot | Käyttäjä osuu ansaan tai putoaa laudalta. |
| Käyttäjä | Normaali käyttäjä |
| Normaali toiminta | 1. Elämä menetetään<br><br>2. Käynnistetään animoitu räjähdys ja soitetaan räjähdysääni. |

| | |
|---|---|
| | 3. Pallo siirtyy viimeisimpään spawnauspisteeseen |
| Poikkeustilanteet | 1. Jos elämiä on nolla, siirrytään kuolema-sceneen.<br>2. Kuolema scene tulostaa ruudulle tiedon kuolemisesta ja näyttää scoren. Huom. Kuolemalla ei pääse hiscore-listalle. |
| Tulos | Elämän menetys tai lopullinen kuolema. |
| Toteutussuunnitelma | Palloon on liitetty koodi, joka vertaa osuiko se ansaan. |

### 6.5 Ulkoiset liittymät

Pelissä ei ole muita liittymiä kuin kosketusnäyttö ja kiihtyvyysanturin lukeminen.

### 6.6 Muut ominaisuudet

Suorituskyky on riittävä jopa halpamallisella puhelimella. Käytettävyys on hyvä, vaikkakin varsinkin menun grafiikat vaativat skaalautumista. Toipuminen virhetilanteista on huono, koska peli kaatuessaan sulkeutuu. Ylläpidettävyys on kehittäjälle helppo, mutta asiaan perehtymättömälle huono, koska eri peliobjekteja ja koodeja on linkitetty toisiinsa monella tapaa. Koodeihin voisi kommentoida paremmin mikä toinen koodinpätkä lukee niissä olevia muuttujia ja, että mihin peliobjektiin koodinpätkä on linkitetty. Peli vie tällä hetkellä käännettynä noin 40 Mb, joten sen siirtäminen ei ole mikään mahdoton asia.

# 7 TOTEUTUS

Tämän otsikon alle on kerätty joitain oleellisimpia koodeja. Ensiksi osia PlayerMoverista, joka sisältää pallon liikkumisen mahdollistavan koodin tabletille ja myös PC:lle testausta varten. Skriptin luokan nimen täytyy olla sama kuin sen tiedoston nimen.

```
public class PlayerController : MonoBehaviour
{
```

Alla oleva funktio ajetaan skenen käynnistyessä kerran, ja se määrittää paikan johon siirrytään kuollessa.

```
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        startPoint = transform.position;
    }
```

FixedUpdatea ajetaan kerran joka freimin aikana, eli reaaliaikaiset toiminnot tulevat tänne. Normaalisti käytetään pelkkää Update()-funktiota, mutta tuo RigidBody vaatii tuon FixedUpdate()-funktion.

Kiihtyvyysanturiin perustuva ohjaus:

```
        float moveX = Input.acceleration.x;
        float moveY = Input.acceleration.y;
        Vector3 movement = new Vector3(moveX, 0.0f, moveY);
        rb.AddForce(movement * speed);
```

Näppäimistöohjaus PC:llä testaamista varten:

```
        if (Input.GetKey(KeyCode.RightArrow)) {
           rb.AddForce(32, 0, 0);
        }
        if (Input.GetKey(KeyCode.LeftArrow))
```

```
        {
            rb.AddForce(-32, 0, 0);
        }

        if (Input.GetKey(KeyCode.UpArrow))
        {
            rb.AddForce(0, 0, 32);
        }

        if (Input.GetKey(KeyCode.DownArrow))
        {
            rb.AddForce(0, 0, -32);
        }
    }
```

Alla oleva koodi on PlayerKiller-skriptistä, joka huolehtii törmäyksien tutkimisesta ja elämien menettämisesät. Kun pallo osuu johonkin, katsotaan onko se Hazard, eli ansa tai vastaava. Jos on, niin vähennetään elämiä yhdellä HUD-skriptissä, käynnistetään räjähdysanimaatio, siirretään pallo aloituspisteeseen ja soitetaan räjähdysääni.

```
    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Hazard")
        {
            HUD.Lives = HUD.Lives - 1;
Instantiate(Explosion, GameObject.Find("AmigaBall").transform.position,
Quaternion.identity);
            transform.position = startPoint;
            GetComponent<AudioSource>().Play();
```

Jos elämät HUD-skriptissä on nolla, kutsutaan Die()-funktiota.

```
if (HUD.Lives == 0)
{
    Die();
}
```

Alla oleva koodi on Infinite-loop kentässä olevan viestin näyttämiseen ja pallon siirtämiseen alkuun.

```
if (other.gameObject.tag == "Respawn")
{
    HUD_ForLoop.MessageInfiniteLoop = 1;
    transform.position = startPoint;
}
```

Kuolema-funktio lataa kuolema-skenen ja sen pitäisi tallentaa pelaajan score tiedostoon.

```
    void Die()
    {
        SceneManager.LoadScene("Death");
        PlayerPrefs.SetInt("Hiscore", HUD.Score);
    }
}
```

DontRotateCamera on käytössä kentissä, joissa kamera seuraa palloa. Koska kamera täten liitettynä pyrkii pyörimään pallon mukana, täytyy yksi sen akseleista lukita.

```
gameObject.transform.position = BallCamera.transform.position + offset;
```

OnGUI()-funktio huolehtii tekstin ja kuvien tulostamisesta ruudulle. Käytännössä siis kaikki, mikä ei ole pelissä kolmiulotteista hoidetaan tällä. Alla valittu tekstiin käytettävä fontti ja tulostettu Score käyttäen kyseistä fonttia. Viimeinen rivi tulostaa kuvan ruudulle, tässä tapauksessa VAMK-logo. Numerot 200 ja 64 ovat kuvan koko pikseleinä ja tässä olisi itseasiassa järkevämpää käyttää jotain Screen.width / 10 ja Screen.heigh / 10.

```
void OnGUI()
    {

        Font     Digital7     =     (Font)Resources.Load("Fonts/Digital-7",
typeof(Font));

        GUIStyle StyleHUD = new GUIStyle();

        StyleHUD.normal.textColor = Color.green;

        StyleHUD.fontSize = 64;

        StyleHUD.font = Digital7;

        GUI.Label(new Rect(10, 10, 200, 200), "Score: " + Score, StyleHUD);

        GUI.DrawTexture(new  Rect(Screen.width  -  210,  Screen.height  -  74,
200, 64), VAMK);
```

# 8 YHTEENVETO

Labyrinth on peli, jonka alustana toimii Android-tabletit ja sen ohjaus tapahtuu laitetta kallistamalla (kiihtyvyysantureita lukemalla). Se on toteutettu Unity 3D-pelimoottorilla ja kaikki siihen sisältyvät koodit on tehty itse. Osa peliobjekteista on ladattu Unity 3Dn Asset Storesta kun taas ääniä ja tekstuureja on ladattu Internetistä. Musiikit ovat ystävieni tekemiä, skriptauskielenä pääasiallisesti C#. Päättötyön aihe on siis itse valittu ja sen tarkoitus on oppia ohjelmoimista ja Unity 3Dn käyttöä.

Idea peliin on saatu ollessani Vaasan ammattikorkeakoulun toisella luokalla ja perustuu isoäitini vanhaan puusta valmistettuun mekaaniseen labyrintti-peliin. Pelissä siis ohjataan palloa labyrintissä vältellen ansoja ja keräten pisteitä. Pelin voi läpäistä pääsemällä lipun luokse. Koulun toivomuksesta peliin lisättiin myös muutama ohjelmointiaiheinen puzzle.

Pelin perusidea toimii hyvin, joskin sen kaupallistamiseen on vielä matkaa ja kehittyneemmät ominaisuudet vaativat vielä säätöä. Lopullisen pelin julkaisen Google Play Storessa.

# LÄHTEET

Amigapallon historia. Wikipedian verkkosivut. Viitattu 2.11.2017.
https://en.wikipedia.org/wiki/Amiga